Data Encryption
Fast & Secure

The algorithm for the Data Encryption Standard runs too slow on most micros, but simpler methods have not provided secure encryption. This program solves this problem by being both fast and secure.

---

by Harry J. Smith

The Data Encryption Standard, DES (1), though normally considered a very secure form of encryption, has a very complicated algorithm (2) and runs very slow when implemented on a micro computer. The program CRYPT (3) as implemented on most systems that use the UNIX operating system is quite fast, but is not a very secure form of data encryption. The data encryption program CRYP presented here attempts to be even more secure than DES by using a larger and more random key, and at the same time is reasonably fast.

The program TNT (4) is a good example of a data encryption method that has attempted to be both fast and secure but it too is about 10 to 20 times more complicated, as measured by program execution time, than the method presented here. This forces TNT to be written in assembly language to be practical. CRYP was written in a high order language and is about 4.5 times faster, exclusive of I/O time, than TNT written in assembly language.

Another article (5) contained encryption programs that were fast and were written in a high order language, but they used the same encryption key over and over again for the same input key. If one enciphered message were ever compromised then any other message which used the same input key would no longer be secure with this method.

Basic Method Used

An alphanumeric key, input by the operator of the program, is converted into nine 16-bit seeds for nine different random number generators. The output of the nine generators are combined to generate a random sequence of bits of about 3.55 * 10**44 bits long (approximately 2**148 bits). This sequence of bits is then used as a pseudo infinite key and is exclusive-ORed with the bits of the data file to produce the data for the enciphered file.

Processing the Key

The characters of the key input by the operator are converted to a standard form containing only 63 different characters. Lower case letters are converted to upper case and control characters are converted to the special characters and the numbers. The space character is changed to a '/' so the standard form will never contain a space. This is done because it may be difficult on some systems to input a space character in an argument on the command line.

A key of 24 characters is needed. If the input key is longer than 24 characters, it is hashed into a 24 character key. For the first 24 characters, each character is increased by the sum of all characters that precede it. For the 25 character on, each character is increased by the sum of all characters that follow it. Then characters in the same position modulo 24 are then added together to make the key 24 characters long. The summing of characters is done arithmetically modulo 256.

If the key is shorter than 24 characters, trailing '/' are appended to make it 24 characters. The 24-character key is also converted to the standard form. Only the 6 least significant bits of each of the 24 key characters are used to generate the seeds for the random number generators.

Continuation File

In order to make this method an infinite key encryption system (4) a continuation file, CRY.CON, is maintained by the program. This file contains one record of 18 bytes from the random number generators. Each time a file is enciphered, the continuation file is read, its 18 bytes are exclusive-ORed with the 18 bytes of the bits from the encryption key to produce the starting seeds for the 9 random number generators. The continuation file is also output as the first 18 bytes of the enciphered file. After the input file is enciphered the continuation file is updated by reseeding the random number generators with the 18 bytes of the continuation file and the next 4096 bytes from the generators are stored in a pool of random numbers in reverse order. Each 16 bits stored is the sum of 9 random integers, one from each generator, truncated to the lower 16 bits. The first 18 bytes of this pool is then written to the continuation file.

There is an option in CRYP to initialize the continuation file. When this option is selected the encryption key is used directly to seed the random number generators and 18 bytes output from the generators, as described above, are stored in the continuation file. The key used to initialize the continuation file should not be the same key used to encrypt data files.

The use of the continuation file does not cause each file enciphering to start up with the pseudo infinite key exactly where it left off, but causes it to start up at a random point in its extremely long periodic cycle. It is impractical to try to start up where the previous file left off because the pool of 2048 random integers, described shortly, would have to be save and protected between runs. Starting up at a random point and reinitializing the pool for each file is a better method.

The Random Number Generators

The nine random number generators were chosen very carefully and have been tested to ensure they each produce the proper number of random integers before they cycle and start reproducing the same sequence all over again. Four of the generators are Congruential generators (6) and five of them are Shift-register generators (7). This was done to prevent the problems that can arise when only one type of generator is used (7). The generators were chosen so their

periods would be relatively prime to each other (65536, 65535, 65537, 65521, 65519, 65497, 65479, 65449 and 65447 respectively). This was done to produce a very long resulting period in the pseudo infinite key. The pseudo infinite key is generated in 16-bit increments by a combination of exclusive-ORing and arithmetic addition of the output of the nine random number generators.

The Pseudo Infinite Key

After the seeds of the 9 generators are established, each generator is called once and the sum of the 9 integers generated, truncated to 16 bits, is used to initialize rr, the running random integer. This rr is normally updated by adding, and truncating to 16 bits, the next output of the next generator, cycling the generators 0 through 8 and repeating. Thus, only one call to one generator is needed to get 16 more bits for the basic random bit sequence.

Next the pool of 2048 random integers is initialized. This is done by storing consecutive values of rr generated as just described. After this rr is recomputed by calling each generator once and summing the 9 integers generated as before. This makes rr independent from any integer in the pool.

Now we are ready to start generating bits of the pseudo infinite key. The next 16 bits of this key is always generated by:

1) Use the lower 11 bits of rr as a relative index into the pool.

2) Use the integer at this position in the pool as the next 16 bits of the pseudo infinite key.

3) Update rr by adding the output of the next generator to rr and keep the lower 16 bits.

4) Replace the integer just used in the pool with the exclusive-OR of itself and rr (do not change rr).

This process causes the pseudo infinite key to be a very complicated function of the output of the 9 random number generators, but is a simple process to program and execute.

One criticism of this method might be that it is a substitution cipher only and that a combination of a transposition and substitution produces the strongest cipher schemes. The normal transposition process is what makes the TNT program run so slow. In a sense CRYP does have a transposition process but it is a transposition in the pseudo infinite key instead of in the text.

Process Overview

In order to understand the method better, outlines of the three options of the program are given:

Encipher Process-

1) Process key from operator.
2) Read continuation file.
3) Copy continuation file to output file.

4) Initialize random number seeds.
5) Fill random number pool.
6) Read, encrypt, write input file to output file.
7) Restore random number seeds from continuation file.
8) Build and write random record to continuation file.

Decipher Process-

1) Process key from operator.
2) Read first 18 bytes of input file.
3) Initialize random number seeds.
4) Fill random number pool.
5) Read, encrypt, write input file to output file.

Initialize Continuation file-

1) Process key from operator.
2) Initialize random number seeds.
3) Build and write random record to continuation file.

Timing Data
A file of 128K bytes of all zeros was generated using DEBUG, and COPY in the following way:

```
A:\>DEBUG
-FCS:100,L,8000,0
-RCX
CX xxxx
:8000
-RBX
BX xxxx
:0
-NA:Z
-W
Writing 08000 bytes
-Q

A:\>COPY /B Z+Z+Z+Z Z128K
Z
Z
Z
Z
1 File(s) copied
```

This file was then enciphered on a hard disk drive in 6.2 seconds and then deciphered in 9.4 seconds. The system used was a 386 AT clone running MS DOS 3.3 at a clock rate of 8 MHz.

Randomness Testing
The 128k file of all zero described above was enciphered four different times and a Chi-square test applied to each enciphered file. The Chi-squared statistic generated was 232.8, 294.8, 304.5 and 225.9. For a Chi-square test with 255 degrees of freedom as this test is, the Chi-square statistic has a median value of 254.3 and should be between 200.6 and 316.9, 99% of the time and between 219.0 and 293.2, 90% of the time.

Twenty-four Chi-square tests were run on other encipherings of the all zero file, taking 5120 byte blocks of data per test. The maximum Chi-square was 291.3, the minimum Chi-square was 198.8 and the average was 254.7. On these 24 runs the mean, second, third, and fourth moments of the distribution of the value of the bytes were also calculated with no significant deviation from their theoretical values.

Operating the program
This program is written in C and all operator input comes from the DOS command line. The format of the command to execute CRYP can be found by executing CRYP with no parameters. The output to the screen in this case is:

```
CRYP - A data encryption system written in C, with pipes
Version 6.00, 1992-11-15, 1400 hours
Copyright (c) 1987-1992 by author: Harry J. Smith

  usage: CRYP key [D/I] [<infile] [>outfile]
    D = Decipher
    I = Init CRY.CON file using key and exit
    default = Encipher and update CRY.CON file
```

The key parameter is a string of any length the system supports, but probably not more than 72 characters. If the key is the only parameter given, then the standard input is encrypted and output to the standard output. The [D/I] parameter is an optional field. When it is present and is the single character D or d the input is deciphered instead of enciphered. When the [D/I] parameter is the single character I or i the continuation file is initialized. The same key that is used to encipher a file must be used to decipher it, but should not be used to initialize the continuation file.

A file can be enciphered twice with the same or different keys and then deciphered by deciphering twice using the keys in the reverse order. This double encryption is not recommended since CRYP is very secure with only one encryption.

Choosing a key
If you are serious about protecting the information contained in your file, the encryption key should be chosen carefully, and of course it also needs to be protected. A key of FEBRUARY is a poor choice because if a few characters can be determined the rest can be guessed. A short key such as ASDF is bad because if the key space is searched in lexicographic order the key will be found in less than 2**24 tries.

A good scheme for choosing a key is to make up a phrase that makes no special sense but is at least 48 characters long. When this key is converted to the 24-character key the characters will look random. For example, the eight input keys:

1) YOU/SHOULD/MAKE/YOUR/KEYS/AT/LEAST/48/CHARACTERS
2) TRY/TO/MAKE/YOUR/KEYS/EASY/TO/REMEMBER/BUT/HARD/TO/BREAK
3) THE/WIND/BLOWS/THE/SNOW/FALLS/IT/WILL/NOT/BE/HOT
4) A/NEST/IS/A/HOME/IF/YOU/ARE/A/BIRD/BUT/I/LIKE/MINE
5) ROSE/HAS/HAD/TOO/MUCH/SAID/I/HAVE/NOTHING/TO/ADD

```
   6) ROPE/IT/CAN/BE/LONG/IT/CAN/BE/SHORT/BUT/NOT/FOR/ME
   7) EAGLE/WHAT/A/BIRD/IF/I/COULD/FLY/WOULD/BE/ONE/TOO
   8) BOOK/READ/ONE/TO/ME/READ/ONE/MYSELF/BUT/NOT/TO/YOU
```

produce the following eight 24-character keys:

```
   1) B>DRQJM](YTR_2(4%36G3**1
   2) U/1"@N)C422T+58;(>/9D2%"
   3) IKOR]SRMH[PVAHHNGXXIR2!!
   4) *J!!%88?I&##)#<282,2/*P2
   5) :@OEKD]OHK]S33:@!'M<55GD
   6) Y:W-Z^#_ZLHCVY3+KKC>8F&7
   7) $'99::+'/4,^XNSVXB\STXXG
   8) ._*'14,4%ORZ0]\WWVLGJ[MB
```

respectively. The long form of the key is easier to remember and easier to type, the short form is more random and harder to crack.

How Secure is the Method?
The search space is about 2**144 keys. An extremely powerful computer would be needed to search and find the key in a timely manner. If 10**14 processors could each test 10**14 different keys per second it would take about 10**7 years to find a randomly selected key.

If there is a flaw in this method it would be that a chunk of the pseudo key, if it were exposed, could be used to determine the value of the seeds used to produce it. This seams totally impossible because the process of generating the pseudo infinite key appears to be quite irreversible, but this is not proven. Thus, it is still necessary to change the input key periodically.

Pascal version
An interactive user interface version of this program, CRYU, was developed using TURBO Pascal. Tests were performed to insure that a file can be enciphered with one of the programs and then deciphered with the other program. The running time of the two programs are essentially the same.

C Named files version
A C version CRYN that uses files names on the command line instead of pipes was also developed. The format of the command to execute CRYN can be found by executing CRYN with no parameters. The output to the screen in this case is:

```
   CRYN - A data encryption system written in C, named files
   Version 6.00, 1992-11-15, 1400 hours
   Copyright (c) 1987-1992 by author: Harry J. Smith

   usage: CRYN key [infile outfile [D]]
     key only => Init CRY.CON file using key and exit
         no D => Encipher and update CRY.CON file
            D => Decipher
```

In all versions of the of the program, if a key is given which is a file name, then the first line of the file is used as the key.

Notes

1. Data Encryption Standard, U. S. Department of Commerce, National Bureau of standards, FIPS Publication 46, 1977 January 15.
2. Katzan, H. The Standard Data Encryption Algorithm. New York: Petrocelli Books, 1977.
3. Kernighan, B., and Plauger, P. Software Tools. Reading, Mass.: Addison-Wesley, 1976.
4. Thomas, J., and Thersites, J. "Designing a File Encryption System", Dr.Dobb's Journal (August 1984): 44-53.
5.Scacchitti, F. E., "The Cryptographer's Toolbox", Dr. Dobb's Journal (May 1986): 58-64.
6. Knuth, D. E. The Art of Computer Programming, vol. 2, Seminumerical Algorithms. Reading, Mass.: Addison-Wesley, 1968.
7. Ralston, A. Encyclopedia of Computer Science, First Edition, "Random Number Generation", New York: Van Nostrand Reinhold, 1976.
8. Stout, R. B., "S-CODER for Data Encryption" Dr. Dobb's Journal (Jan 1990): 52-58.

Harry J. Smith
19628 Via Monte Dr.
Saratoga, CA 95070
(408) 741-0406